OPERATING SYSTEM ABSTRACTION AND PROTECTION LAYER

[0001] This application is a continuation of U.S. patent application Ser. No. 09/859209, filed on May 16, 2001, entitled *Operating System Abstraction and Protection* Layer, the entirety of which is incorporated herein by reference as if fully set forth.

[0002] The present invention relates to computer software, and more particularly to operating system software.

BACKGROUND OF THE INVENTION

[0003] In many environments, but particularly in environments where an application is delivered via a network, the most important feature is an ability to run applications on the fly, without a complex installation. Typically, in certain prior art systems, great pains were taken to modify a client system to appear as if a program was installed, or to actually install the software itself, and then back out these modifications to restore the original configuration. In doing this, multiple problems present themselves: conflicts between an application and the computer's current configuration, multiple instances of the same or different applications, complexity of the back out process requires an application to be put through a rigorous process to ensure all of its modifications can be accounted for, and the use of shared files and system components by multiple applications complicates back out and the installation process.

SUMMARY OF THE INVENTION

[0004] The present invention provides a system for creating an application software environment without changing an operating system of a client computer, the system comprising an operating system abstraction and protection layer, wherein said abstraction and protection layer is interposed between a running software application and said operating system, whereby a virtual environment in which an application may run is provided and application level interactions are substantially removed. Preferably, any changes directly to the operating system are selectively made within the context of the running application and the abstraction and protection layer dynamically changes the virtual environment according to administrative settings. Additionally, in certain embodiments, the system continually monitors the use of shared system resources and acts as a service to apply and remove changes to system components.

[0005] Thus, for example, in embodiments within Windows-based operating systems, and wherein all operations to the Windows Registry are through the Win32 API, the system preferably provides a means for hooking functions, whereby each time said functions are invoked another function or application intercepts the call, and the system most preferably hooks each appropriate API function to service a request whether made by an application run from a server or if made by an application against a configuration key being actively managed.

[0006] In other preferred embodiments of the present invention, additional functionality is provided, such as those embodiments wherein the operating system abstraction and protection layer manages the integration of multiple instances of an application by recognizing how many instances of an application are running, and in such embodiments most preferably it also avoids making changes on startup and shutdown unless there is only one application instance running. In this embodiment it is also possible to support multi-user operating systems in which multiple instances of an application can be running on behalf of different users.

[0007] Thus, the operating system abstraction and protection layer presents an environment to an application that appears to be an installation environment without performing an installation, whereby a "pseudo installation" is created in which all of the settings are brought into a virtual environment at the time the application runs. Or in the case of an installed application, acts to dynamically modify the behavior of the application at run-time. Preferred embodiments provide a means for preventing information on the client computer from interfering or modifying the behavior of an application, and most preferably provide a means for dynamically changing the virtual environment according to administrative settings. As mentioned above, in certain embodiments it will be possible to have more than one instance of a single software application running on the same client computer, even if it was not originally authored to do so. In such embodiments, shared, controlled contexts are provided in which at least two of said instances of a single application share one or more virtual settings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] In the drawing,

[0009] **FIG. 1** is a block diagram schematic showing the relative relationship of the present invention, an operating system and a software application;

[0010] **FIG. 2** is a block diagram schematic showing two applications running with private contexts and services;

[0011] **FIG. 3** is a block diagram schematic showing two applications running while the operating system provides shared view of the system resources; and

[0012] **FIG. 4** is a block diagram schematic showing an operating system guard and subsystems.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0013] Referring now to **FIG. 1**, there is illustrated a block diagram schematic showing the relative relationship of the present invention, an operating system and a software application. Preferred embodiments of the present invention provide an operating system abstraction and protection layer **100** denominated an "Operating System Guard." Internally, many operating systems **10** provide fault domains to protect applications **50** from affecting each other when run. However, shared system resources and many other operating system features allow this protection domain to be compromised. An operating system abstraction and protection layer **100** will provide an additional, programmatically controlled barrier between applications **50** to remove most application level interactions. Disposed between the application **50** and operating system **10** the operating system abstraction and protection layer **100** selectively allows changes directly to the operating system **10**, versus containing the change within the context of the running application. For one example, in Windows-based systems, all operations to the Windows Registry are typically done through the Win32